**DOCKER**

1. Docker Networking
2. Explore the current setup and identify the number of networks that exist on this system.

docker network ls

NETWORK ID NAME DRIVER SCOPE

89303905eb33 bridge bridge local

a9ddd20e58a2 host host local

38672f05baf9 none null local

1. What is the ID associated with the bridge network, what is the subnet configured on bridge network?

docker inspect bridge

[

{

"Name": "bridge",

"Id": **"89303905eb33c64522921179bc4ef096cf7d15358f3a680c9e43b2f99647b3dc",**

"Created": "2025-04-10T03:07:47.694724941Z",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": null,

"Config": [

{

**"Subnet": "172.12.0.0/24",**

"Gateway": "172.12.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {},

"Options": {

"com.docker.network.bridge.default\_bridge": "true",

"com.docker.network.bridge.enable\_icc": "true",

"com.docker.network.bridge.enable\_ip\_masquerade": "true",

"com.docker.network.bridge.host\_binding\_ipv4": "0.0.0.0",

"com.docker.network.bridge.name": "docker0",

"com.docker.network.driver.mtu": "1500"

},

"Labels": {}

}

]

1. We just ran a container named alpine-1. Identify the network it is attached to.

docker inspect alpine-1

"NetworkSettings": {

"Bridge": "",

"SandboxID": "",

"SandboxKey": "",

"Ports": {},

"HairpinMode": false,

"LinkLocalIPv6Address": "",

"LinkLocalIPv6PrefixLen": 0,

"SecondaryIPAddresses": null,

"SecondaryIPv6Addresses": null,

"EndpointID": "",

"Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"IPAddress": "",

"IPPrefixLen": 0,

"IPv6Gateway": "",

"MacAddress": "",

"Networks": {

**"host": {**

"IPAMConfig": null,

"Links": null,

"Aliases": null,

"MacAddress": "",

"NetworkID": "a9ddd20e58a296eb7739da0c571e8bf1ed78beb32306f2230c9eaf16e90077f5",

"EndpointID": "08f6e31b4bd10abdb5765b864c0fca11046ad17e9c9888bbc8d8dede44f33bf5",

"Gateway": "",

"IPAddress": "",

"IPPrefixLen": 0,

"IPv6Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"DriverOpts": null,

"DNSNames": null

}

}

}

}

]

1. Run a container named alpine-2 using the alpine image and attach it to the none network.

docker run --name alpine-2 --network none -d alpine

2bc16faa46daf77aacde3762e9b7e61b455268a43fb2aadf33b74da0a4c4c4e0

1. Create a new network named wp-mysql-network using the bridge driver. Allocate subnet 182.18.0.0/24. Configure Gateway 182.18.0.1

docker network create -d bridge --subnet 182.18.0.0/24 --gateway 182.18.0.1 wp-mysql-network

74811e1b5f052eb93580eb1c15ec5b0eb0c0b3c4ba131f0978239aeefeadcb6c

1. Deploy a mysql database using the mysql:5.6 image and name it mysql-db. Attach it to the newly created network wp-mysql-network

Set the database password to use db\_pass123. The environment variable to set is MYSQL\_ROOT\_PASSWORD.  
  
  
docker run --name mysql-db --network wp-mysql-network -e MYSQL\_ROOT\_PASSWORD=db\_pass123 -d mysql:5.6  
  
G. Deploy a web application named webapp using the kodekloud/simple-webapp-mysql image. Expose the port to 38080 on the host.  
  
The application makes use of two environment variable:  
1: DB\_Host with the value mysql-db.  
2: DB\_Password with the value db\_pass123.  
Make sure to attach it to the newly created network called wp-mysql-network.

Also make sure to link the MySQL and the webapp container.

docker run --name webapp -p 38080:8080 -e DB\_Host=mysql-db -e DB\_Password=db\_pass123 --network wp-mysql-network --link mysql-db:mysql-db kodekloud/simple-webapp-mysql

\* Serving Flask app "app" (lazy loading)

\* Environment: production

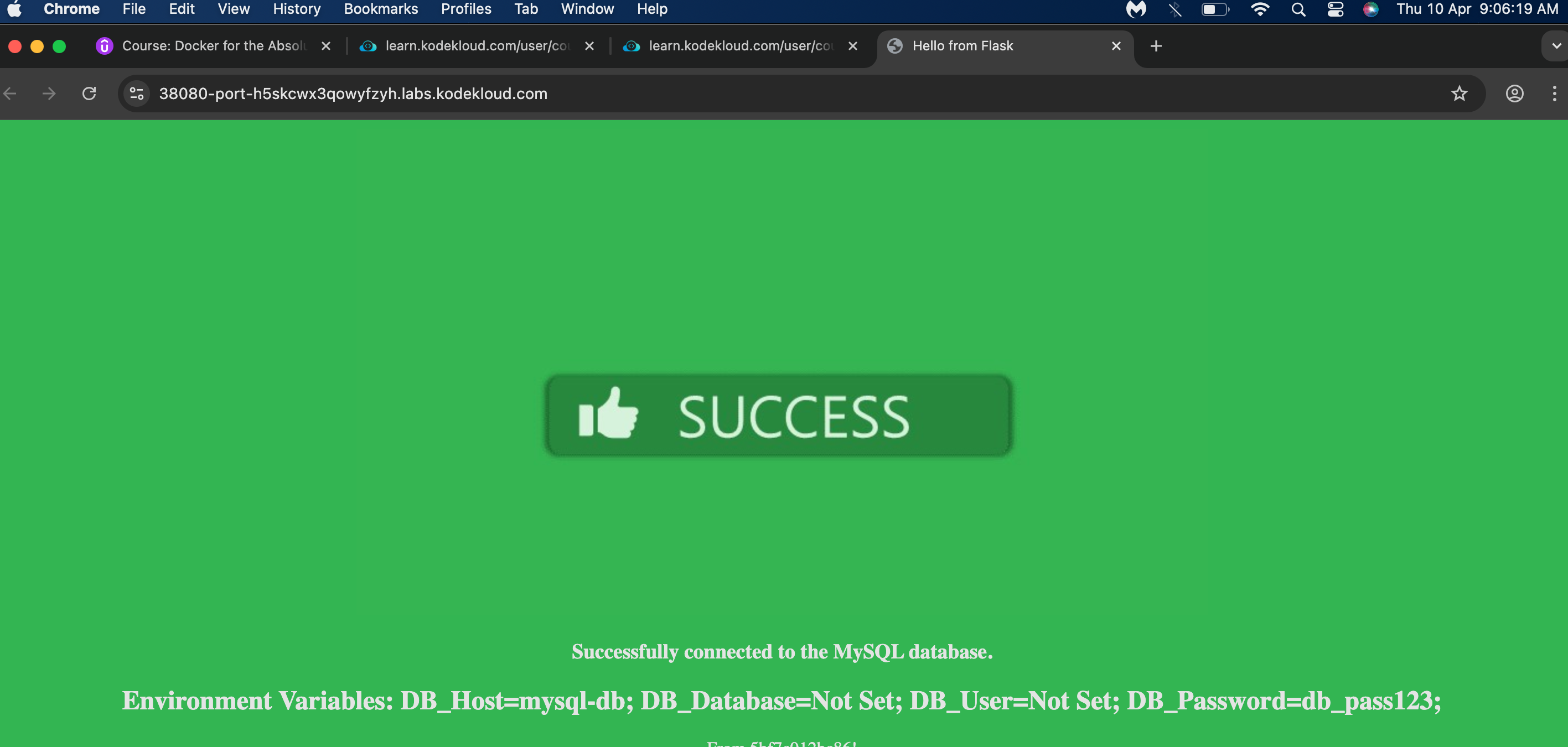
WARNING: Do not use the development server in a production environment.

Use a production WSGI server instead.

\* Debug mode: off

\* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)

192.168.139.232 - - [10/Apr/2025 03:36:11] "GET / HTTP/1.1" 200 -

1. 192.168.139.232 - - [10/Apr/2025 03:36:13] "GET /static/img/success.jpg HTTP/1.1" 200 -  
     
     
     
   2. Docker Storage

A. What location are the files related to the docker containers and images stored?

/var/lib/docker  
  
B. Run a mysql container again, but this time map a volume to the container so that the data stored by the container is stored at /opt/data on the host.

Use the same name : mysql-db and same password: db\_pass123 as before. Mysql stores data at /var/lib/mysql inside the container.

docker run --name mysql-db -v /opt/data:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=db\_pass123 mysql